第42卷 第5期
2022年10月
桂林电子科技大学学报
Journal of Guilin University of Electronic Technology
Vol. 42, No. 5
Oct. 2022

# 基于最短长链接优先选择的VLSI阵列重构算法

莫福浩，钱俊彦

（桂林电子科技大学 计算机与信息安全学院，广西 桂林 541004）

**摘　要**：为了提高超大规模集成逻辑阵列的重构效率，提出一种基于最短长链接优先选择原则的改进算法。该算法从阵列2端分别构建逻辑列，直到2条逻辑列相交，则停止构建逻辑列。在2条相交逻辑列作为边界的局部区域内，以从上到下的方式寻找每行长链接最短的处理器单元，所选取的处理单元用于构建局部最优的逻辑列。基于上述操作，利用分治思想，将新获得的逻辑列作为新的局部区域的边界，依次迭代获得新的局部最优逻辑列。最后，将所得到的局部最优逻辑列连接起来，即可获得最终的目标阵列。通过与现有的重构算法的比较分析，验证了算法的高效性。仿真结果表明，在保证逻辑阵列规模不变的条件下，相较于现有的重构算法，该算法能够有效减少阵列重构过程中处理器的访问数，并能在一定程度上降低重构的运行时间，提高逻辑阵列的重构效率。

**关键词**：重构；算法；处理阵列；容错；高效率

**中图分类号**：TP301.6　　**文献标志码**：A　　**文章编号**：1673-808X(2022)05-0371-05

# Reconstruction algorithm for VLSI array with the priority selection of the shortest long interconnect

MO Fuhao, QIAN Junyan

(School of Computer and Information Security, Guilin University of Electronic Technology, Guilin 541004, China)

**Abstract**: In order to improve the reconstruction efficiency of the very large-scale integrated logic array, an improved algorithm based on the principle of first selecting the shortest and long interconnect is proposed. This improved algorithm constructs logical columns from both ends of the array until the two logical columns intersect, then stops constructing logical columns. In the local area with two intersecting logical columns as the boundary, the processor unit with the shortest link length of each row is searched from top to bottom, and the selected processing unit is used to construct the locally optimal logical column. Based on the above operations, using the idea of divide and conquer, the newly obtained logical column is used as the boundary of the new local area, and the new local optimal logical column is obtained in turn. Finally, connect the obtained local optimal logical columns to obtain the final target array. Through comparison and analysis with existing reconstruction algorithms, the efficiency of the proposed algorithm is verified. The final simulation results show that under the condition that the scale of the logic array remains unchanged, compared with the existing reconstruction algorithm, this improved algorithm can effectively reduce the number of processor accesses during the array reconstruction process, and to a certain extent, the running time of the reconstruction is reduced, and the reconstruction efficiency of the logic array is improved.

**Key words**: reconfiguration; algorithm; VLSI array; fault tolerance; high-efficiency



随着超大规模集成（VLSI）技术和晶圆规模集成（WSI）技术的发展，集成系统得以在芯片上构建。然而，随着VLSI阵列中处理单元（PE）密度的增加，VLSI阵列发生故障的概率也随之增加，从而导致处理器的计算能力和稳定性受到影响，进而影响整个集成系统的正常工作。因此，需要采用容错技术对阵列

**收稿日期**：2021-03-06
**基金项目**：国家自然科学基金(61562015)；广西自然科学基金(2018GXNSFDA138003)
**通信作者**：钱俊彦(1973－)，男，教授，博士，研究方向为软件工程、程序验证与测试、容错技术。E-mail：qjy2000@gmail.com.
**引文格式**：莫福浩，钱俊彦. 基于最短长链接优先选择的VLSI阵列重构算法[J]. 桂林电子科技大学学报，2022，42(5)：371-375.

进行重构来提高阵列的可靠性。

网络拓扑结构的 VLSI 重构技术主要有冗余法和降阶法[1] 2 种。在冗余法中，集成系统存在部分备用的处理单元，当系统中出现故障处理单元时，可用备用的处理单元对故障的处理单元进行替换；在降阶法中，系统的元件均以统一的方式进行处理，无额外的备用元件，这种方法使用阵列中尽可能多的无故障处理单元来重构目标阵列，从而提高无故障处理单元的使用效率。

在过去的几十年间，二维处理器阵列的重构策略被广泛研究。Kuo 等[2] 提出了二维处理器阵列下 3 种路由重构方案：1）行、列旁路；2）行旁路和列重路由；3）行、列重路由，同时证明了阵列的重构问题是 NP 完全问题。Low 等[3] 用 GCR 算法构建包含所选行的最大目标阵列。基于片上网络阵列，Xiang 等[4] 提出了一种新的用于全网单播组播核心测试的传输容错路由算法，并基于重叠虚拟网络策略，提出了片上网络无死锁自适应路由方案；此外，Xiang 等[5] 提出了一种高性能自适应路由策略，有效降低了芯片网络的功耗；基于重复转向模型，Cai 等[6] 提出了基于无死锁自适应方案的三维片上网络路由算法。重构方案主要从时间、规模、内部链接长度等方面对阵列进行优化。在降低阵列重构时间方面，Wu 等[7] 用并行算法减少阵列的构建时间；Qian 等[8] 提出了一种智能搜索算法，有效提升了阵列的重构效率；Qian 等[9] 提出了一种可满足最大功率效率 VLSI 阵列重构的有效方法。在减少阵列长连接方面，Srikanthan 等[10] 通过重构紧耦合目标阵列优化了阵列的长连接；肖汉鹏等[11] 设计了紧耦合阵列的整数规划模型；Qian 等[12] 提出了一种构造高性能 VLSI 子阵列的有效多重最短增广路径算法；黄弼胜等[13] 利用网络流模型减少阵列的连接长度。在扩大阵列规模方面，王意萍等[14] 基于整数规划思想获取了最大规模阵列；Qian 等[15] 在行和列重路由方案下重构了 VLSI 子阵列的数学模型；Ding 等[16] 提出了在行和列重路由下重构二维网格连接 VLSI 子阵列的灵活方案；Qian 等[17-18] 利用网络流算法对阵列进行优化，并提出了一种新的算法，扩大了阵列的规模。白章顺等[19] 提出了一种容错处理器阵列的重构抽象模型；胡佳等[20] 提出了一种高性能阵列重构的 SAT 描述模型；Wu 等[21] 将阵列的研究从二维阵列向三维阵列延伸；Jiang 等[22] 提出了一种高效的重构算法，降低了三维阵列中的连接长度；Jiang 等[23] 在取消限制补偿距离的条件下提出的 FLX 算法进一步扩大了获取的阵列规模，并在 FLX 算法的基础上，基于分治策略的思想，提出了 RIL 算法，减少了阵列中的长链接数。然而，由于 RIL 算法在重构逻辑列的过程中需要对局部区域内所有的节点进行访问，在一定程度上影响了阵列的重构时间，降低了重构效率。因此，为了减少构建阵列过程中对节点的访问数，加快阵列的重构速度，在 RIL 算法的基础上，提出了一种基于最短长链接优先选择原则的重构（acceleration of RIL，简称 ACRIL）算法。



## 1　二维阵列结构与问题描述

### 1.1　阵列结构

在二维 VLSI 阵列中，制造产生的原始阵列用 $H$ 表示，阵列大小为 $m\times n$，其中 $m$、$n$ 分别为原始阵列的行数和列数。假设 $p$ 为原始阵列的故障密度，$0\leqslant p\leqslant 1$，则原始阵列中出现故障的处理单元的个数 $N$ 表示为 $N=(1-p)\times m\times n$。故障的处理单元表示不能对数据进行处理或者从其他处理单元获取数据的处理单元。原始阵列经过重新配置后获得的阵列称为逻辑阵列，用 $T$ 表示，阵列大小为 $m'\times n'$（$m'\leqslant m$，$n'\leqslant n$），逻辑阵列中不包含故障的处理单元。原始阵列和逻辑阵列中的行（列）分别称为物理行（列）和逻辑行（列）。原始阵列的物理结构如图 1 所示。

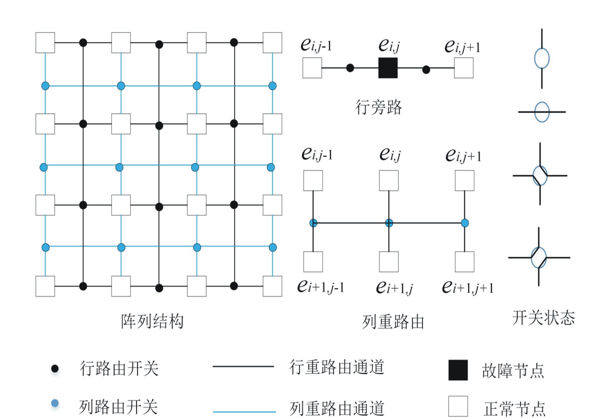



**图 1　VLSI 阵列体系结构**

每个开关有 4 种链路状态。每个处理单元都可通过改变路由开关的链接状态来改变处理单元之间的连接方式。$e_{i,j}$、$e'_{i,j}$ 分别为物理阵列、逻辑阵列中处于位置（$i$，$j$）的处理单元，其中 $i$ 为处理单元的行号，$j$ 为处理单元的列号。原始阵列有 2 种基本的重构方案，分别是行旁路方案和列重路由方案。在行旁路方案中，$e_{i,j-1}$ 可直接通过改变内部开关连接状态，绕过故障的处理单元 $e_{i,j}$，直接与 $e_{i,j+1}$ 相连，同理，列旁路方案的定义与之类似。在列重路由方案中，$e_{i+1,j}$ 为故障处理单元，可通过外部开关直接连

接到 $e_{i,j'}$,其中 $|j-j'|<d$,$d$ 为补偿距离,通常限制为 1,从而能够有效保证硬件实现低功耗。同理,行重路由方案的定义与之类似。

根据补偿距离的定义,每个处理单元的上相邻节点集合和下相邻节点集合分别记为 $A_-(u)$、$A_+(u)$,且定义如下:

**定义 1** 对于行中的每个无故障处理单元,

1)$A_-(u)=\{v:v\in \mathbf{R}_{i-1},v$ 为无故障处理单元,且 $|\mathrm{col}(u)-\mathrm{col}(v)|\leqslant 1\},2\leqslant i\leqslant m$。

2)$A_+(u)=\{v:v\in \mathbf{R}_{i+1},v$ 为无故障处理单元,且 $|\mathrm{col}(u)-\mathrm{col}(v)|\leqslant 1\},1\leqslant i\leqslant m-1$。

3)对于任意的 $v\in A_+(u)(A_-(u))$,当 $\mathrm{col}(u)-\mathrm{col}(v)=-1$ 时,$v$ 称为 $u$ 的左下(上)邻接处理单元;当 $\mathrm{col}(u)-\mathrm{col}(v)=0$ 时,$v$ 称为 $u$ 的中下(上)邻接处理单元;当 $\mathrm{col}(u)-\mathrm{col}(v)=1$ 时,$v$ 称为 $u$ 的右下(上)邻接处理单元。

一个二维逻辑阵列有 6 种可能的链路方式,根据链路使用的开关数,可分为短连接与长连接。只使用一个开关来连接目标阵列中的处理单元的链路方式称为短连接,而使用 2 个开关的链路方式称为长连接。

**定义 2** 长连接数量最少的阵列称为高性能逻辑阵列(HPTA)。

### 1.2 问题描述

在行旁路和列重路由条件下对二维处理器阵列的重构问题进行研究,研究内容包括寻找高性能的无故障目标阵列。

**问题 1** 给定一个规模大小为 $m\times n$ 的带有故障节点的网状连接处理器阵列,找到长链接数目最小的无故障高性能目标阵列。

## 2 ACRIL 算法介绍

RIL 算法通过运用分治策略的思想解决了不限制补偿距离条件下的问题 1。为了优化 RIL 算法的效率,采用 ACRIL 算法,通过减少对无故障节点的访问次数,重构局部优化逻辑列,以降低逻辑阵列的重构时间。

ACRIL 算法的求解结果如图 2 所示。该算法主要分为 3 个子过程:Up_Process 表示第一个公共处理单元到第一行的子逻辑列求解过程;Down_Process 表示最后一个公共处理单元到最后一行的子逻辑列求解过程;Shest_Process 表示公共处理单元之间的子逻辑列求解过程。图 2 中的数字表示处理单元的长连接。

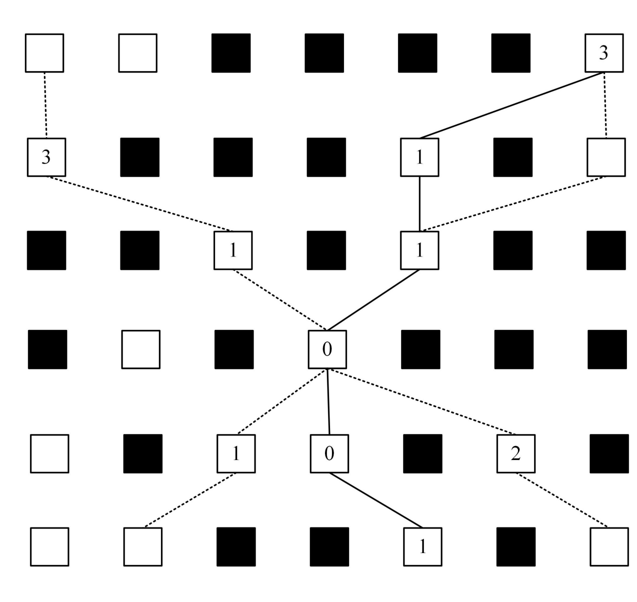


图 2 ACRIL 算法的求解结果

ACRIL 算法的整体思路如下:

1)在一次迭代过程中,找到左右 2 条逻辑列的交点,并将交点放入交点集合中。

2)利用智能搜索算法分别求解 Up_Process、Down_Process 及 Shest_Process 三个子过程,每个子过程获得一段路径。

3)将这 3 个子过程获得的路径进行合并,得到一条完整的路径,此路径即是所要求的一条逻辑列。

4)将新获取的逻辑列作为新的边界,继续重构逻辑列,重复 1)、2)、3)过程。

**算法 1** ACRIL 算法

输入:$m\times n$ 的原始阵列。

输出:$m\times k$ 的目标阵列。

1)根据主阵列,依次分别从左至右和从右至左的方式重构逻辑列。

2)直到从 2 个方向构建的逻辑列连接到相同的节点,形成公共区域。在逻辑列的公共区域构建具有最短长连接的逻辑列。

寻找最优逻辑列的伪代码如下:

```
while 交点集合不为空 do
  for node u ∉ 第一行元素 do
    启发式搜索算法寻找 Up_Process 的路径 P_1;
for node u ∉ 最后一行元素 do
    启发式搜索算法寻找 Down_Process 的路径 P_2;
  if 2 个交点相邻 do
    直接加入路径;
else
    启发式搜索算法思想寻找 Shest_Process 的路径 P_3;
```

将3个部分获得的路径合并形成一条完整的路径 $P=P_1\cup P_2\cup P_3$。

3)以新构建的逻辑列作为新的边界,重复步骤1)、2),直到所有的逻辑列构建完毕。

Shest_Process 子过程主要运用启发式搜索思想进行求解,其求解过程描述如下:

输入:2个公共节点 $S$、$T$。

输出:$S$、$T$ 之间的一段路径 $P$。

1)设置2个列表,分别为开放列表与关闭列表。

2)遍历节点,计算当前节点的权值,根据情况将节点放入相应的列表中;当节点已位于开放列表时,重新计算权值。

3)当开放列表为空时,表示子过程执行完毕。求解子过程的伪代码如下:

```
开放列表:只包含 S;
关闭列表:空集;
while 开放列表不为空 do
  获取开放列表中权值最小的节点作为当前节点;
  if 当前节点为 T then
    return S 和 T 之间的路径 P;
  将当前节点放入关闭列表中;
  while 当前节点的下邻接集合不为空 do
  获取下邻接节点 u;
  if u ∈ 开放列表 do
   重新计算 u 的权值;
  if u ∈ 关闭列表 do
   不用执行任何操作;
  else
   计算 u 的权值,将 u 放入开放列表;
  从下邻接集合中去除 u。
```

Up_Process 和 Down_Process 子过程的求解方式与子过程 Shest_Process 类似。

## 3 实验与分析

为验证 ACRIL 算法的有效性,用 C++语言实现了该算法,还原了现有的重构算法 RIL,并将2种重构算法进行对比。为保证实验数据具有可比性和可靠性,2种重构算法均在相同的实验环境下运行。执行程序的实验配置环境为 Inter® Core™ 3.10 GHz 的 CPU 和 8 GiB RAM,操作系统为 Windows 10。

RIL 算法与 ACRIL 算法的访问节点数的对比如图3所示。图3中,在64×64规模下的原始阵列的错误率分别设置为1%、5%、10%、15%。从图3可看出,在错误率为1%的原始阵列中,RIL 算法对阵列中的节点访问数为5 091,而 ACRIL 算法对阵列中的节点访问数为3 989;在错误率为10%的原始阵列中,RIL 算法对阵列中的节点访问数为5 191,而 ACRIL 算法对阵列中的节点访问数为3 573。2种重构算法的实验数据比较结果表明,相较于现有的重构算法 RIL,ACRIL 算法能在一定程度上减少阵列在重构逻辑列的过程中对节点的访问数,提高阵列的重构效率。

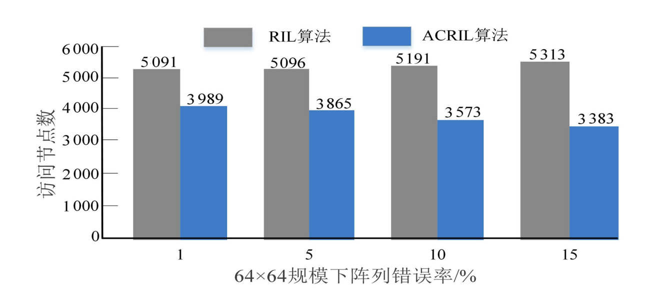



**图3 RIL 与 ACRIL 算法的访问节点数对比**

RIL 算法与 ACRIL 算法性能指标如表1所示。从表1可看出,两者在相同错误率条件下都可获取相同规模的目标逻辑阵列,且相比于 RIL 算法,ACRIL 算法的阵列重构时间更短,这表明阵列从故障状态恢复的效率更高。例如,在错误率为1%、规模为32×32的原始阵列中,RIL 算法的重构时间为674 ms,而 ACRIL 算法的重构时间为574 ms,重构效率提高了14.83%;在错误率为5%,规模为48×48的原始阵列中,RIL 算法的重构时间为956 ms,而 ACRIL 算法的重构时间为821 ms,重构效率提高了14.13%。这表明 ACRIL 算法的运行性能明显优于 RIL 算法。

**表1 RIL 与 ACRIL 算法的性能指标**

| 原始阵列规模 | 错误率/% | 目标阵列规模 | 重构时间/ms | | 重构效率/% |
|---|---|---|---|---|---|
| | | | RIL | ACRIL | |
| 32×32 | 1 | 32×31 | 674 | 574 | 14.83 |
| | 5 | 32×27 | 641 | 554 | 13.57 |
| | 10 | 32×23 | 614 | 529 | 13.84 |
| 48×48 | 1 | 48×45 | 1 037 | 926 | 10.70 |
| | 5 | 48×42 | 956 | 821 | 14.13 |
| | 10 | 48×36 | 870 | 782 | 10.11 |
| 64×64 | 1 | 64×61 | 2 281 | 2 078 | 8.90 |
| | 5 | 64×55 | 2 118 | 1 926 | 9.10 |
| | 10 | 64×50 | 1 995 | 1 801 | 9.72 |

## 4 结束语

基于现有的重构算法,提出了一种通过减少处理单元的访问次数,进一步提高原始阵列重构速度的方

案。实验结果表明,与现有的重构算法相比,该算法显著减少了处理单元的访问次数,减少了重构时间,降低了运行时的性能开销。本研究未考虑在构建阵列过程中逻辑列的选择对阵列规模的影响。今后的研究将着重于考虑新的改进算法,为可降阶的 VLSI 阵列的重构提供更好的性能优化。

**参考文献:**


[1] ZHANG Li. Fault-tolerant meshes with small degree[J]. IEEE Transactions on Computers, 2002, 51(5): 553-560.

[2] KUO S Y, CHEN I Y. Efficient reconfiguration algorithms for degradable VLSI/WSI arrays[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1992, 11(10): 1289-1300.

[3] LOW C P, LEONG H W. On the reconfiguration of degradable VLSI/WSI arrays[J]. IEEE Transaction on Computer-Aided Design of Integrated Circuits and System, 1997, 16(10): 1213-1221.

[4] XIANG Dong, CHAKRABARTY K, FUJIWARA H. Fault-tolerant unicast based multicast for reliable network-on-chip testing[J]. ACM Transactions on Design Automation of Electronic Systems, 2018, 23(73): 1-73.

[5] XIANG Dong, PAN Qunyang. Low-power and high-performance adaptive routing in on-chip networks[J]. CCF Transactions on High Performance Computing, 2019, 1(6): 92-110.

[6] CAI Yuan, XIANG Dong, JI Xiang. A deadlock-free adaptive 3D network-on-chips routing algorithm with repetitive turn concept[J]. IET Communications, 2020, 14(11): 1783-1792.

[7] WU Jigang, JIANG Guiyuan, SHEN Yuze, et al. Parallel reconfiguration algorithms for mesh-connected processor arrays[J]. The Journal of Supercomputing, 2014, 69(2): 610-628.

[8] QIAN Junyan, ZHOU Zhide, ZHAO Lingzhong, et al. Accelerating reconguration for VLSI arrays with A-star algorithm[J]. IEEJ Transactions on Electrical and Electronic Engineering, 2018, 13(10): 1511-1519.

[9] QIAN Junyan, CHEN Cong, ZHAO Lingzhong, et al. An efficient method for reconfiguring power efficient VLSI array with maximum satisfiability[J]. IEEJ Transactions on Electrical and Electronic Engineering, 2018, 13(5): 770-776.

[10] WU Jigang, SRIKANTHAN T, JIANG Guiyuan, et al. Constructing sub-arrays with short interconnects from degradable VLSI arrays[J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(4): 929-938.

[11] 肖汉鹏,钱俊彦. 紧耦合处理器阵列重构的整数规划模型[J]. 桂林电子科技大学学报, 2018, 38(1): 61-64.

[12] QIAN Junyan, HUANG Bisheng, DING Hao, et al. An efficient multiple shortest augmenting paths algorithm for constructing high performance VLSI subarray[J]. Integration the VLSI Journal, 2021, 75: 63-72.

[13] 黄弼胜,钱俊彦. 基于网络流的高效 VLSI 子阵列重构[J]. 桂林电子科技大学学报, 2019, 39(6): 466-470.

[14] 王意萍,钱俊彦. 基于整数规划的最大目标阵列重构算法[J]. 桂林电子科技大学学报, 2017, 37(6): 458-462.

[15] QIAN Junyan, WANG Yiping, CHANG Liang, et al. A mathematical model for reconfiguring VLSI subarrays under row and column rerouting[J]. IEEE Access, 2017, 5: 23912-23919.

[16] DING Hao, QIAN Junyan, ZHAO Lingzhong, et al. Flexible scheme for reconfiguring 2D mesh-connected VLSI subarrays under row and column rerouting[J]. Journal of Parallel and Distributed Computing, 2021, 151: 1-12.

[17] QIAN Junyan, ZHOU Zhide, GU Tianlong. Optimal reconfiguration of high-performance VLSI subarray swith network flow[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3575-3587.

[18] QIAN Junyan, DING Hao, XIAO Hanpeng. Efficient reconfiguration algorithm with flexible rerouting schemes for constructing 3D VLSI sub-arrays[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(1): 267-271.

[19] 白章顺,钱俊彦. 容错处理器阵列的重构抽象模型[J]. 桂林电子科技大学学报, 2017, 37(1): 36-39.

[20] 胡佳,钱俊彦,周志德. 高性能阵列重构的 SAT 描述模型[J]. 桂林电子科技大学学报, 2017, 37(1): 40-43.

[21] JIANG Guiyuan, WU Jigang, HA Yajun, et al. Reconfiguring three-dimensional processor arrays for fault-tolerance: hardness and heuristic algorithms[J]. IEEE Transactions on Computers, 2015, 64(10): 2926-2939.

[22] JIANG Guiyuan, WU Jigang, SUN Jizhou, et al. Reducing the interconnection length for 3d fault-tolerant processor arrays[C]//International Conference on Algorithms and Architectures for Parallel Processing. Berlin, German: Springer, 2014: 497-510.

[23] JIANG Guiyuan, WU Jigang, SUN Jizhou, et al. Flexible rerouting schemes for reconfiguration of multiprocessor arrays[J]. Journal of Parallel and Distributed Computing, 2014, 74(10): 3026-3036.


编辑:张所滨